

## CLAIMS

What is claimed is:

- 1 1. A method comprising:
  - 2 detecting that a guest operating system attempts to access a region
  - 3 occupied by a first portion of a virtual machine monitor (VMM) within a first
  - 4 address space; and
  - 5 relocating the first portion of the VMM within the first address space to
  - 6 allow the guest operating system to access the region previously occupied by
  - 7 the first portion of the VMM.
- 1 2. The method of claim 1 wherein the first portion of the VMM includes a
  - 2 set of VMM code and data structures that are architecturally required to
  - 3 reside in the first address space.
- 1 3. The method of claim 1 wherein the first portion of the VMM includes a
  - 2 set of trap handlers and an interrupt-descriptor table (IDT).
- 1 4. The method of claim 1 further comprising:
  - 2 dividing the VMM into the first portion and a second portion;
  - 3 creating the first address space associated with the guest operating
  - 4 system;
  - 5 creating a second address space associated with the VMM;

6 locating the second portion of the VMM in the second address space  
7 associated with the VMM; and  
8 mapping the first portion of the VMM into the first address space and  
9 the second address space.

1 5. The method of claim 1 further comprising:  
2 receiving control over an event initiated by the guest operating system  
3 when the event may potentially cause an address space conflict between the  
4 guest operating system and the VMM.

1 6. The method of claim 5 wherein receiving control further comprises:  
2 setting access rights of the section occupied by the first portion of the  
3 VMM to a more privileged level than a privilege level associated with the  
4 guest operating system; and  
5 receiving a trap caused by an attempt of the guest operating system to  
6 access a hardware resource having a higher privilege level than the privilege  
7 level associated with the guest operating system.

1 7. The method of claim 6 further comprising:  
2 determining that the trap can be handled by the first portion of the  
3 VMM;  
4 executing code associated with the trap; and  
5 returning control over the event to the guest operating system.

1 8. The method of claim 6 further comprising:  
2 determining that the trap should be handled by the second portion of  
3 the VMM;  
4 delivering the trap to the second portion of the VMM;  
5 passing control over the event to the guest operating system after code  
6 associated with the trap was executed by the second portion of the VMM.

1 9. The method of claim 1 wherein relocating the first portion of the VMM  
2 further comprises:  
3 finding an unused region within the first address space; and  
4 re-mapping the first portion of the VMM into the unused region.

1 10. The method of claim 1 wherein relocating the first portion of the VMM  
2 further comprises:  
3 determining that no unused region exists within the first address  
4 space;  
5 selecting a random region within the first address space;  
6 copying content of a memory located at the random region to the  
7 second address space; and  
8 re-mapping the first portion of the VMM into the random region.

1 11. The method of claim 10 further comprising:

2 receiving control over an event initiated by the guest operating system,  
3 the event corresponding to an attempt of the guest operating system to access  
4 the content of the memory previously located at the random region; and  
5 accessing the copied content of the memory in the second address  
6 space.

1 12. The method of claim 11 further comprising periodically relocating the  
2 first portion of the VMM to random regions within the first address space  
3 until finding a region that is infrequently accessed.

1 13. An apparatus comprising:  
2 a first address space associated with a guest operating system;  
3 a second address space associated with a virtual machine monitor  
4 (VMM); and  
5 a virtual machine kernel to detect that the guest operating system  
6 attempts to access a region occupied by a first portion of the VMM within the  
7 first address space and to relocate the first portion of the VMM within the first  
8 address space to allow the guest operating system to access the region  
9 previously occupied by the first portion of the VMM.

1 14. The apparatus of claim 13 wherein the first portion of the VMM  
2 includes a set of VMM code and data structures that are architecturally  
3 required to reside in the first address space.

1 15. The apparatus of claim 13 wherein the first portion of the VMM  
2 includes a set of trap handlers and an interrupt-descriptor table (IDT).

1 16. The apparatus of claim 13 wherein the virtual machine kernel is to  
2 divide the VMM into the first portion and the second portion, to locate the  
3 second portion of the VMM in the second address space associated with the  
4 VMM, and to map the first portion of the VMM into the first address space  
5 and the second address space.

1 17. The apparatus of claim 13 wherein the virtual machine kernel is to  
2 receive control over an event initiated by the guest operating system when the  
3 event may potentially cause an address space conflict between the guest  
4 operating system and the VMM.

1 18. The apparatus of claim 13 wherein the virtual machine kernel is to  
2 receive control by setting access rights of the section occupied by the first  
3 portion of the VMM to a more privileged level than a privilege level  
4 associated with the guest operating system, and by receiving a trap caused by  
5 an attempt of the guest operating system to access a hardware resource  
6 having a higher privilege level than the privilege level associated with the  
7 guest operating system.

1 19. The apparatus of claim 18 wherein the virtual machine kernel is to  
2 further determine that the trap can be handled by the first portion of the

3 VMM, to execute code associated with the trap, and to return control over the  
4 event to the guest operating system.

1 20. The apparatus of claim 18 wherein the virtual machine kernel is to  
2 further determine that the trap should to handled by the second portion of the  
3 VMM, to deliver the trap to the second portion of the VMM, and to pass  
4 control over the event to the guest operating system after code associated  
5 with the trap was executed by the second portion of the VMM.

1 21. The apparatus of claim 13 wherein the virtual machine kernel is to  
2 relocate the first portion of the VMM by finding an unused region within the  
3 first address space and re-mapping the first portion of the VMM into the  
4 unused region.

1 22. The apparatus of claim 13 wherein the virtual machine kernel is to  
2 relocate the first portion of the VMM by determining that no unused region  
3 exists within the first address space, selecting a random region within the first  
4 address space, copying content of a memory located at the random region to  
5 the second address space, and re-mapping the first portion of the VMM into  
6 the random region.

1 23. The apparatus of claim 13 wherein the virtual machine kernel is to  
2 receive control over an event initiated by the guest operating system, the  
3 event corresponding to an attempt of the guest operating system to access the

- 4 content of the memory previously located at the random region, and to access  
5 the copied content of the memory in the second address space.

1 24. The apparatus of claim 13 wherein the virtual machine kernel is to  
2 periodically relocate the first portion of the VMM to random regions within  
3 the first address space until finding a region that is infrequently accessed.

1  
1 25. A system comprising:  
2 a memory to include a first address space associated with a guest  
3 operating system and a second address space associated with a virtual  
4 machine monitor (VMM); and  
5 a processor, coupled to the memory, to detect that the guest operating  
6 system attempts to access a region occupied by a first portion of the VMM  
7 within the first address space and to relocate the first portion of the VMM  
8 within the first address space to allow the guest operating system to access  
9 the region previously occupied by the first portion of the VMM.

1 26. The system of claim 25 wherein the first portion of the VMM includes a  
2 set of VMM code and data structures that are architecturally required to  
3 reside in the first address space.

1 27. The system of claim 25 wherein the first portion of the VMM includes a  
2 set of trap handlers and an interrupt-descriptor table (IDT).

1 28. A computer readable medium that provides instructions, which when  
2 executed on a processor, cause said processor to perform operations  
3 comprising:  
4 detecting that a guest operating system attempts to access a region  
5 occupied by a first portion of a virtual machine monitor (VMM) within a first  
6 address space; and  
7 relocating the first portion of the VMM within the first address space to  
8 allow the guest operating system to access the region previously occupied by  
9 the first portion of the VMM.

1 29. The computer readable medium of claim 28 comprising further  
2 instructions causing the processor to perform operations comprising:  
3 finding an unused region within the first address space; and  
4 re-mapping the first portion of the VMM into the unused region.

1 30. The computer readable medium of claim 28 comprising further  
2 instructions causing the processor to perform operations comprising:  
3 determining that no unused region exists within the first address  
4 space;  
5 selecting a random region within the first address space;  
6 copying content of a memory located at the random region to the  
7 second address space; and  
8 re-mapping the first portion of the VMM into the random region.